Joint Transmission Scheduling and Congestion Control for Adaptive Video Streaming in Small-Cell Networks

D. Bethanabhotla, G. Caire and M. J. Neely

Abstract

We consider the optimal design of a scheduling policy for adaptive video streaming in wireless "Small-Cells" networks. We formulate the problem as a network utility maximization, and we observe that it naturally decomposes into two subproblems: admission control and transmission scheduling. The resulting algorithms are simple and suitable for distributed implementation. The admission control decisions involve each wireless user choosing the quality of the video "chunk" asked for download, based on the local network congestion in its neighborhood. This is compatible with the current video streaming technology based on the DASH protocol over TCP connections. The transmission scheduling policy takes on the form of a local maximization of the weighted sum rate at each small cell base station, where the weights are given by the queue backlogs. We also propose a method to locally estimate the delay with which the video chunks are delivered, such that each user can decide its pre-buffering time at the beginning of a streaming session or re-buffering time in the case of a "stall" event (empty playback buffer) during a streaming session. Through simulations, we evaluate the performance of the proposed algorithm under realistic assumptions for a small-cell wireless network. Our results show that the scheme is robust to arbitrary (per-sample path) network state variations due, for example, to random streaming activity and user mobility.

Index Terms

Adaptive Video Streaming, Small-Cells Wireless Networks, Scheduling, Admission Control, Adaptive Pre-Buffering Time.

All authors are with the Ming Hsieh Department of Electrical Engineering, Viterbi School of Engineering, University of Southern California, Los Angeles CA 90089. Email: `bethanab, caire, mjneely@usc.edu`

I. Introduction

We consider the problem of joint transmission scheduling and congestion control for adaptive video streaming in a wireless network formed by number of small cell base stations serving wireless users over a given geographic coverage area [1], [2]. We formulate a Network Utility Maximization (NUM) problem in the framework of Lyapunov Optimization [3], and derive algorithms for joint transmission scheduling and congestion control inspired by the *drift plus penalty* approach [3].

Excellent surveys on various problem formulations in the NUM framework can be found in [4]–[6]. Initial work in NUM focused on networks with static connectivity and time-invariant channels. One of the first applications of NUM was to show that Internet congestion control in TCP implicitly solves a NUM problem where the variant of TCP dictates the exact shape of the utility function [4], [6]. This framework has been extended to wireless ad-hoc networks with time varying channel conditions (see [7], [8] and the references therein) and to peer-to-peer networks in [9]. Recent work on adaptive video scheduling in wireless networks appears in [10], [11].

Video streaming differs from file downloading since the duration of the download is much longer than the delay with which the playback begins with respect to the time at which the request is made. Hence, the video file is divided into segments referred to as "chunks", typically of video playback duration ≈ 0.5s. These chunks are requested sequentially, and the video quality of such segments can be dynamically adjusted depending on the local channel conditions (e.g., cell congestion, fading, pathloss changes due to mobility). After an initial pre-buffering delay, where the requesting user fills its playback buffer with an ordered sequence of chunks, the playback starts while the user is still requesting the remaining chunks. The playback process consumes the chunks at fixed playback rate (e.g., one chunk per 0.5s). In contrast, the number of *ordered* chunks per unit time entering the playback buffer is a random variable, due to the fact that the transmission resources are dynamically allocated by the scheduling policy, and that the network state (to be formally defined later on) may be time-varying. Notice that chunks must be ordered sequentially in order to be useful for video playback. If chunks go through different queues in the network and are affected by different delays, it may happen that already received chunks with higher order number cannot be used for playback until the missing chunks with lower order number are also received. Hence, the playback buffer input process is dominated by the largest delay with which the requested chunks are delivered. The pre-buffering time should be adaptively determined on the basis of an estimate of such largest delay, for each streaming user. We define the "stall" event as the event that a chunk is not in the buffer when it is needed for playback. In this case, several practical options are

possible. For example, the playback may jump to the chunk of lowest order number available in the buffer. Alternatively, the playback may stop until the missing chunk is received. Also, when a stall event occurs, the user may decide to stop playback and dedicate some time to *re-buffering*, i.e., building up a sufficiently large number of ordered chunks in the buffer, before re-starting the playback.

In our network model, wireless *users* place video streaming requests and wish to download sequences of video chunks, corresponding to the desired video files, from neighboring dedicated nodes referred to as *helpers*. User requests are served in a single-hop transmission (we do not consider multi-hop relaying in this work). The helpers may be small cell base stations [2], connected to some video content delivery network (CDN) through some wired backbone, or dedicated wireless video streaming nodes with local caching capacity, disseminated in the coverage area for the specific purpose of creating a "wireless CDN" [12], [13]. Also, the helpers may be other user terminals with cached video files, operating in device-to-device mode [9], [14].

For the problem at hand, as a consequence of the NUM formulation, we obtain an elegant decomposition of the optimal solution into two separate subproblems, namely, transmission scheduling and admission control, which interact through the queue lengths. The subproblems are solved by distributed dynamic policies requiring only local queue length information at each network node. In particular, the transmission scheduling decisions are carried out in a decentralized fashion by the *helpers*, while the admission control decisions are carried out in a decentralized fashion by the *users*. It is interesting to notice that the independent choices made by every user in deciding the quality of the video chunk to request for download at every given time is compatible with the current technology based on client-driven *Dynamic Adaptive Streaming over HTTP* (DASH) for video on demand (VoD) systems [15], [16].

Motivated by realistic system parameters, we assume that the time and frequency selective wireless channel fading coherence time $\times$ bandwidth product is small with respect to the number of signal dimensions spanned by the transmission of a video chunk (see Section II for typical values). This implies that the rate scheduling decisions in the transmission scheduling policy can be based on the *ergodic* achievable rate region of the underlying physical layer, where averaging is with respect to the small-scale channel fading, conditionally on the pathloss coefficients. This approach has the advantage of decoupling the scheduling decisions at the chunk level from the fine-grain physical layer details, such as power control and adaptive coding and modulation, which may be implemented at the physical layer at the time scale of the transmission time-frequency slots, independently and transparently with respect to the proposed scheduling scheme. In contrast, other system parameters such as the distance dependent path loss or the user activity (alternating idle/streaming state) may evolve at the same time scale of the video

chunks, yielding non-ergodic dynamics. For example, a typical YouTube video streaming session spans between 4 to 10 min, which is sufficient for a user moving at walking speed to significantly change its relative distances with respect to surrounding helpers. Also, we have to consider that a coded video file is formed by chunks whose statistics may change with time. Correspondingly, variable bit rate (VBR) video coding [17] yields a rate-quality tradeoff function that may change with time (i.e., with the chunk index). We address these non-stationary and non-ergodic dynamics of the system parameters by providing performance guarantees that hold for *arbitrary sample paths*, using the approach developed in [3], [18].

The rest of this paper is organized as follows. In Section II, we describe our network and requests model for video streaming in a small-cell heterogeneous wireless network scenario, and discuss the key underlying assumptions. In Section III, we formulate the problem in the NUM framework and derive a distributed dynamic scheduling algorithm for its solution using the *drift plus penalty* approach. In Section IV, we provide performance guarantees for any arbitrary sample path evolution of the network state and recover as a special case the corresponding guarantees when the network state is assumed to vary in an i.i.d manner. Section V illustrates our proposed policy for adaptive pre-buffering and re-buffering that is able to cope with the fact that, since different chunks of the same video streaming sessions go through different helpers transmission queues, the sequence of chunks received at the playback buffer may be out of order and some chunks may be missing at the required playback time. Finally, a realistic system simulation scenario is provided in Section VI. We investigate the effect of the key parameters regulating the proposed scheduling policy and affecting the tradeoffs of the relevant performance metrics, such as pre-buffering time, fraction of time spent in *re-buffering* mode, and the quality levels of the downloaded chunks. Our results show that the proposed algorithm dynamically adapts its decisions in response to variations in large scale path-loss coefficients due to mobility, to variations due to user activity (i.e., when users alternate idle times of random duration between successive streaming sessions), and to variations of the source coding rate across video chunks due to VBR video coding.

II. SYSTEM MODEL

We consider a discrete, time-slotted wireless network with multiple user stations and multiple helper stations. The network is defined by a bipartite graph $\mathcal{G} = (\mathcal{U}, \mathcal{H}, \mathcal{E})$, where $\mathcal{U}$ denotes the set of users, $\mathcal{H}$ denotes the set of helpers, and $\mathcal{E}$ contains edges for all pairs (h, u) such that there exists a potential transmission link between $h \in \mathcal{H}$ and $u \in \mathcal{U}$. We denote by $\mathcal{N}(u) \subseteq \mathcal{H}$ the neighborhood of user u, i.e., $\mathcal{N}(u) = \{h \in \mathcal{H} : (h, u) \in \mathcal{E}\}$. Similarly, $\mathcal{N}(h) = \{u \in \mathcal{U} : (h, u) \in \mathcal{E}\}$.

Each user $u \in \mathcal{U}$ requests a video file f_u from a library of possible files. As said before, a video file is

formed by a sequence of chunks (e.g., corresponding to a group of pictures (GOP) that are encoded and decoded as stand-alone units [16]). Chunks have a fixed duration, given by $T_{\text{gop}} = (\text{\# frames per GOP})/\eta$, where η is the frame rate (frames per second), and must be reproduced in sequence at the user end. The streaming process consists of transferring chunks from the helpers to the requesting users such that the playback buffer at each user receiver contains the required chunks at the beginning of each chunk playback time. The playback starts after a short pre-buffering time, where the playback buffer is filled in by a determined amount of ordered chunks. We assume that the scheduler time-scale coincides with the chunk interval, i.e., at each chunk interval a scheduling decision is made. Conventionally, we assume a slotted time axis $t = 0, 1, 2, 3 \ldots$, corresponding to epochs $t \times T_{\text{gop}}$. Letting T_u denote the pre-buffering delay of user u, the chunks are downloaded starting at time $t = 0$ and playback starts at time T_u. A stall event for user u at time $t \geq T_u$ is defined as the event that the playback buffer does not contain chunk number $t - T_u$ at slot time t.

The helpers may not have access to the whole video library. For example, if helpers are connected to the core network through a capacity limited wireless link, then it is convenient to cache locally a subset of the possible files and refresh the caches at much slower rate during the traffic off-peaks [12]. In general, we denote by $\mathcal{H}(f)$ the set of helpers that contain file f. Hence, the request of user u for a chunk at a particular slot t can be assigned to any one of the helpers in the set $\mathcal{N}(u) \cap \mathcal{H}(f_u)$. Letting N_{pix} denote the number of pixels per frame, a chunk contains $k = \eta T_{\text{gop}} N_{\text{pix}}$ pixels (source symbols). We assume that each chunk of file f is encoded at a finite number of different quality modes $m \in \{1, \ldots, N_f\}$. This is similar to what is currently implemented in several recent video streaming technologies like Microsoft Smooth Streaming and Apple HTTP Live Streaming [19]. Due to the variable bit-rate nature of video coding, the quality-rate profile may vary from chunk to chunk. We let $D_f(m,t)$ and $kB_f(m,t)$ denote the video quality measure (e.g., see [20]) and the number of bits for file f at chunk time t and quality mode m respectively. A fundamental function of the network controller at every slot time t consists of choosing the quality mode $m_u(t)$ for all requesting users u. The choice $m_u(t)$ renders the choice of the point $(D_{f_u}(m_u(t),t), kB_{f_u}(m_u(t),t))$ from the finite set of quality-rate tradeoff points $\{(D_{f_u}(m,t), kB_{f_u}(m,t))\}_{m=1}^{N_f}$. We let $R_{hu}(t)$ denote the source coding rate (bit per pixel) of chunk t received by user u from helper h. In addition to choosing the quality mode $m_u(t)$ for chunk time t for all requesting users u, the network controller also allocates the source coding rates $R_{hu}(t)$ satisfying

$$\sum_{h \in \mathcal{N}(u) \cap \mathcal{H}(f_u)} R_{hu}(t) = B_{f_u}(m_u(t), t), \quad \forall\, (h, u) \in \mathcal{E}. \tag{1}$$

When $R_{hu}(t)$ is determined, helper h places the corresponding $kR_{hu}(t)$ bits in its transmission queue

Q_{hu}, to be sent to user u within the queuing and transmission delays. Notice that in order to be able to download different parts of the same chunk from different helpers, the network controller needs to ensure that all received bits from the serving helpers $\mathcal{N}(u) \cap \mathcal{H}(f_u)$ are useful, i.e., the union of all received bits spans the base layer up, without overlaps and without gaps. Alternatively, each chunk can be encoded by intra-session *Random Linear Network Coding* [21] such that as long as $kB_{f_u}(m_u(t), t)$ parity bits are collected at user u, the t-th chunk can be decoded and it becomes available in the user playback buffer. Interestingly, our solution derived in Section III yields that even though we allow the possibility of downloading different bits of the same chunk from different helpers, the optimal scheduling policy is such that users download only entire chunks from single helpers, rather than obtaining different parts from different helpers. Thus, it turns out that the assumption of protocol coordination or linear network coding to prevent overlap or gaps of the downloaded bits from different helpers is not needed.

The dynamics of the transmission queues at the helpers is given by:

$$Q_{hu}(t+1) = \max\{Q_{hu}(t) - n\mu_{hu}(t), 0\} + kR_{hu}(t), \quad \forall \, (h, u) \in \mathcal{E}, \tag{2}$$

where n denotes the number of physical layer channel symbols corresponding to the duration T_{gop}, and $\mu_{hu}(t)$ is the channel coding rate (bits/channel symbol) of the transmission from helper h to user u at time t.

We model the point-to-point wireless channel for each $(h, u) \in \mathcal{E}$ as a frequency and time selective underspread [22] fading channel. Using OFDM, the channel can be converted into a set of parallel narrowband sub-channels in the frequency domain (subcarriers), each of which is time-selective with a certain fading channel coherence time. We assume the widely adopted block fading model, where the small scale Rayleigh fading coefficient is constant over time-frequency "tiles" spanning blocks of adjacent subcarriers in the frequency domain and blocks of OFDM symbols in the time domain. For example, in the LTE 4G standard [23], the small scale fading coefficients can be considered as constant over a coherence time interval of 0.5 ms (7 OFDM symbols) and a coherence bandwidth of 12 sub-carriers, corresponding to a bandwidth of $12 \times 15\text{kHz} = 180\text{kHz}$ for a carrier spacing of 15kHz. Such tiles are referred to as *resource blocks*. For a total system available bandwidth of 18MHz (after excluding the guard bands) and a scheduling slot of duration $T_{\text{gop}} = 0.5$s (typical GOP duration), we have that a scheduling slot spans $\frac{0.5 \times 18 \cdot 10^6}{0.5 \cdot 10^{-3} \times 180 \cdot 10^3} = 10^5$ resource blocks, each of which is affected by its own fading coefficient. Even assuming some correlation between fading coefficients, it is apparent that the time-frequency diversity experienced in the transmission of a chunk is *very large*. Thus, it is safe to assume that channel coding over such a large number of resource blocks can achieve the *ergodic capacity* of the underlying fading

channel. [1] For simplicity, in this paper we consider constant power transmission, i.e., the serving helpers transmit with constant and flat power spectral density over the whole system bandwidth, irrespective of the scheduling decisions and of the instantaneous fading channel state.[2] We further assume that every user u when decoding a transmission from a particular helper $h \in \mathcal{N}(u)$ treats the interference from other helpers as noise. Under these system assumptions, the maximum achievable rate at slot time t for link $(h, u) \in \mathcal{E}$ is given by

$$C_{hu}(t) = \mathbb{E}\left[\log\left(1 + \frac{P_h g_{hu}(t)|a_{hu}|^2}{1 + \sum_{h' \neq h} P_{h'} g_{h'u}(t)|a_{h'u}|^2}\right)\right], \tag{3}$$

where P_h is the transmit power of helper h, a_{hu} is the small-scale fading gain from helper h to user u and $g_{hu}(t)$ is the slow fading gain (pathloss) from helper h to user u.

We assume that each helper h serves its neighboring users $u \in \mathcal{N}(h)$ using orthogonal FDMA/TDMA. Therefore, the set of rates $\{\mu_{hu}(t) : u \in \mathcal{N}(h)\}$ is constrained to be in the "time-sharing region" of the broadcast channel formed by helper h and its neighbors $\mathcal{N}(h)$. This yields the transmission rate constraint

$$\sum_{u \in \mathcal{N}(h)} \frac{\mu_{hu}(t)}{C_{hu}(t)} \leq 1, \quad \forall\, h \in \mathcal{H}. \tag{4}$$

Also note that in practice, for a dense deployment of helpers, each helper h serves only a small number of users, i.e., $|\mathcal{N}(h)|$ is small and therefore scheduling at the physical layer for the sake of achieving multi user diversity gains does not provide significant improvements in terms of ergodic rates. Therefore, for simplicity we neglect such a possibility and use the above ergodic rate expression that assumes to schedule users independently of their instantaneous small-scale fading channel realizations, thereby avoiding the need for fast channel state information feedback. It is however assumed that some form of *slow* channel state information feedback exists, consistent with rate adaptation schemes currently implemented in WiFi and LTE [23]. This allows the helpers to adapt their rate such that the rate vector allocated by each helper $h \in \mathcal{H}$ to its served users $u \in \mathcal{N}(h)$ is inside the (generally time-varying) achievable rate region (4) at each time t.

---

[1]In this paper we refer to *ergodic capacity* as the *average* mutual information resulting from Gaussian i.i.d. inputs of the single-user channel from helper h and user $u \in \mathcal{N}(h)$, while treating the signals of all other helpers $h' \neq h$ as noise, where averaging is with respect to the first-order distribution of the small-scale fading, conditionally on the set of slow fading gains. This rate is achievable by i.i.d. Gaussian coding ensembles and approachable in practice by modern graph-based codes provided that the length of a codeword spans a very large number of independent small-scale fading states [24].

[2]This assumption is consistent to a WiFi offloading system, where the helpers coincides with the WiFi access points, and operate at fixed power and over the whole system bandwidth.

The slow fading gain $g_{hu}(t)$ models path loss and shadowing between helper h and user u, and it is assumed to change slowly in time. For a scenario typical of small cell networks, where users are nomadic (e.g., moving at walking speed), the slow fading coefficients change on a time-scale of the order of 10s (i.e., ≈ 20 scheduling slots). This time scale is much slower than the coherence of the small-scale fading, but it is comparable with the duration of the video chunks. Therefore, variations of these coefficients during a streaming session (e.g., due to user mobility) are relevant. We let $\omega(t)$ denote the network state at time t, i.e.,

$$\omega(t) = \{g_{hu}(t), D_{f_u}(\cdot, t), B_{f_u}(\cdot, t) : \forall\, (h, u) \in \mathcal{E}\}.$$

Let $A_{\omega(t)}$ be the set of feasible control actions, dependent on the current network state $\omega(t)$, and let $\alpha(t) \in A_{\omega(t)}$ be a control action, comprising the vectors $\mathbf{R}(t)$ with elements $kR_{hu}(t)$ of video coded bits, $\boldsymbol{\mu}(t)$ with elements $n\mu_{hu}(t)$ of channel coded bits and $m_u(t)\ \forall\ u \in \mathcal{U}$. A control policy for the system at hand is a sequence of control actions $\{\alpha(t)\}_{t=0}^{\infty}$ where at each time t, $\alpha(t) \in A_{\omega(t)}$.

III. Problem Formulation and Control Policy Design

In the proposed NUM problem, the goal consists of designing a control policy which maximizes a concave utility function of the time averaged *video qualities* of the users, subject to keeping the transmission queues at every helper stable. Throughout this work, we use the following notation for the long-term time average expectation of any quantity x:

$$\overline{x} := \lim_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\left[x(\tau)\right] \tag{5}$$

Define $\overline{D}_u := \lim_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\left[D_{f_u}\left(m_u(\tau), \tau\right)\right]$ as the long-term time average of the expected quality of user u, and $\overline{Q}_{hu} := \lim_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\left[Q_{hu}(\tau)\right]$ as the long-term time average of the expected length of the queue at helper h for data transmission to user u assuming temporarily that these limits exist [3]. Let $\phi_u(\cdot)$ be a concave, continuous, and non-decreasing function defining utility vs. video quality for user $u \in \mathcal{U}$. The goal is to solve:

$$\text{maximize} \quad \sum_{u \in \mathcal{U}} \phi_u(\overline{D}_u) \tag{6}$$

$$\text{subject to} \quad \overline{Q}_{hu} < \infty\ \forall\ (h, u) \in \mathcal{E} \tag{7}$$

$$\alpha(t) \in A_{\omega(t)}\ \forall\ t, \tag{8}$$

---

[3] The existence of these limits is assumed temporarily for ease of exposition of the optimization problem (6) – (8) but is not required for the analysis in Section IV.

where constraint (7) corresponds to the *strong stability* condition for all the queues Q_{hu}.

Problem (6) – (8) can be solved using the stochastic optimization theory of [3]. Since it involves maximizing a *function* of time averages, it is convenient to transform it into an equivalent problem that involves maximizing a single time average instead of a function of time averages. Then, the *drift plus penalty* framework of [3] can be applied. This transformation is achieved through the use of auxiliary variables $\gamma_u(t)$ and corresponding virtual queues $\Theta_u(t)$ with buffer evolution:

$$\Theta_u(t+1) = \max\{\Theta_u(t) + \gamma_u(t) - D_{f_u}(m_u(t), t), 0\}. \tag{9}$$

Consider the transformed problem:

$$\text{maximize} \quad \sum_{u \in \mathcal{U}} \overline{\phi_u(\gamma_u)} \tag{10}$$

$$\text{subject to} \quad \overline{Q}_{hu} < \infty \ \forall \ (h, u) \in \mathcal{E} \tag{11}$$

$$\overline{\gamma}_u \leq \overline{D}_u \ \forall \ u \ \in \ \mathcal{U} \tag{12}$$

$$D_u^{\min} \leq \gamma_u(t) \leq D_u^{\max} \ \forall \ u \ \in \ \mathcal{U} \tag{13}$$

$$\alpha(t) \in A_{\omega(t)} \ \forall \ t \tag{14}$$

where $D_u^{\max}$ is a uniform upper bound on the maximum quality $D_{f_u}(N_{f_u}, t)$ and $D_u^{\min}$ is a lower bound on the minimum quality $D_{f_u}(1, t)$, for all chunk times t. Notice that constraints (12) correspond to stability of the virtual queues Θ_u, since $\overline{\gamma}_u$ and $\overline{D}_u$ are the time-averaged arrival rate and the time-averaged service rate for the virtual queue given in (9). We have:

Lemma *1:* Problems (6) – (8) and (10) – (14) are equivalent.

Proof: Let ϕ_1^{opt} and ϕ_2^{opt} be the optimal solutions of problems (6) – (8) and (10) – (14), respectively. Let $\alpha^*(t)$ be a feasible policy[4] that achieves ϕ_2^{opt}. Since $\phi_u(\cdot)$ is concave for all $u \in \mathcal{U}$, by Jensen's inequality, we have

$$\sum_{u \in \mathcal{U}} \phi_u(\overline{\gamma_u^*}) \geq \sum_{u \in \mathcal{U}} \overline{\phi_u(\gamma_u^*)} = \phi_2^{\text{opt}} \tag{15}$$

and since the policy $\alpha^*(t)$ satisfies the constraint (12), $\phi_u(\cdot)$ is non-decreasing for all $u \in \mathcal{U}$, we further have

$$\sum_{u \in \mathcal{U}} \phi_u(\overline{D_u^*}) \geq \sum_{u \in \mathcal{U}} \phi_u(\overline{\gamma_u^*}). \tag{16}$$

---

[4] Note that there is a slight abuse of notation here because we now assume that the policy $\alpha^*(t)$ also includes the decisions $\boldsymbol{\gamma}^*(t) = (\gamma_1^*(t), \dots, \gamma_{|\mathcal{U}|}^*(t))$ in addition to the decisions $\mathbf{R}^*(t)$, $\boldsymbol{\mu}^*(t)$ and $m_u^*(t)$ for all $u \in \mathcal{U}$, as indicated at the end of the previous section.

Notice that since $\alpha^*(t)$ is feasible for the transformed problem (10) – (14), then it also satisfies the constraints of the original problem (6) – (8) and therefore it is feasible for this problem. Thus we conclude that:

$$\phi_{\text{opt}}^1 \geq \sum_{u \in \mathcal{U}} \phi_u(\overline{D_u^*}) \geq \phi_{\text{opt}}^2. \tag{17}$$

Now let $\alpha'(t)$, comprising of the decisions $\mathbf{R}'(t)$, $\boldsymbol{\mu}'$ and $m_u'(t)$ for all $u \in \mathcal{U}$, be an optimal poicy for the original problem, achieving ϕ_1^{opt}. Since $\alpha'(t)$ satisfies the constraints (7), (8), it also satisfies the constraints (11), (14) of the transformed problem. Further, we choose $\boldsymbol{\gamma}'(t) = \overline{\mathbf{D}'}$ for all time t. Such choice of $\boldsymbol{\gamma}'(t)$ together with the policy $\alpha'(t)$ forms a feasible policy for the transformed problem (10) – (14). Therefore:

$$\phi_2^{\text{opt}} \geq \sum_{u \in \mathcal{U}} \overline{\phi_u(\gamma_u')} = \sum_{u \in \mathcal{U}} \phi_u(\overline{D_u'}) = \phi_1^{\text{opt}}. \tag{18}$$

Thus, from (17) and (18), we have $\phi_1^{\text{opt}} = \phi_2^{\text{opt}}$ and by comparing the constraint it is immediate to conclude that an optimal policy for the transformed problem can be directly turned into an optimal policy for the original problem. ∎

Let $\mathbf{Q}(t)$ denote the column vector containing the backlogs of queues $Q_{hu} \ \forall \ (h, u) \in \mathcal{E}$ at time t and let $\boldsymbol{\Theta}(t)$ denote an analogous column vector for the virtual queues $\Theta_u \ \forall \ u \in \mathcal{U}$. Also let $\boldsymbol{\gamma}(t)$ denote the vector with elements $\gamma_u(t) \ \forall \ u \in \mathcal{U}$ and $\mathbf{D}(t)$ denote the vector with elements $D_{f_u}(m_u(t), t) \ \forall \ u \in \mathcal{U}$. The *drift plus penalty* (DPP) policy which solves the problem (10) – (14) is obtained as follows. Let $\mathbf{G}(t) = \left[\mathbf{Q}^\mathsf{T}(t), \boldsymbol{\Theta}^\mathsf{T}(t)\right]^\mathsf{T}$ be the combined vector of queue backlog vectors and define the quadratic Lyapunov function $L(\mathbf{G}(t)) := \frac{1}{2} \mathbf{G}^\mathsf{T}(t) \mathbf{G}(t)$. The one-slot drift of the Lyapunov function at slot t is given by

$$L(\mathbf{G}(t+1)) - L(\mathbf{G}(t))$$
$$= \frac{1}{2} (\mathbf{Q}^\mathsf{T}(t+1)\mathbf{Q}(t+1) - \mathbf{Q}^\mathsf{T}(t)\mathbf{Q}(t)) + \frac{1}{2} (\boldsymbol{\Theta}^\mathsf{T}(t+1)\boldsymbol{\Theta}(t+1) - \boldsymbol{\Theta}^\mathsf{T}(t)\boldsymbol{\Theta}(t))$$
$$= \frac{1}{2} \left[(\max\{\mathbf{Q}(t) - \boldsymbol{\mu}(t), \mathbf{0}\} + \mathbf{R}(t))^\mathsf{T} (\max\{\mathbf{Q}(t) - \boldsymbol{\mu}(t), \mathbf{0}\} + \mathbf{R}(t)) - \mathbf{Q}^\mathsf{T}(t)\mathbf{Q}(t) \right]$$
$$+ \frac{1}{2} \left[(\max\{\boldsymbol{\Theta}(t) + \boldsymbol{\gamma}(t) - \mathbf{D}(t), \mathbf{0}\})^\mathsf{T} (\max\{\boldsymbol{\Theta}(t) + \boldsymbol{\gamma}(t) - \mathbf{D}(t), \mathbf{0}\}) - \boldsymbol{\Theta}^\mathsf{T}(t)\boldsymbol{\Theta}(t) \right]. \tag{19}$$

Note that for $Q \geq 0$, $\mu \geq 0$, $R \geq 0$

$$(\max\{Q - \mu, 0\} + R)^2 \leq Q^2 + \mu^2 + R^2 + 2Q(R - \mu), \tag{20}$$

and for $\Theta \geq 0$, $\gamma \geq 0$ and $D \geq 0$

$$(\max\{\Theta + \gamma - D, 0\})^2 \leq (\Theta + \gamma - D)^2 = \Theta^2 + (\gamma - D)^2 + 2\Theta(\gamma - D). \tag{21}$$

Using inequalities (20) and (21) in (19), we have

$$L(\mathbf{G}(t+1)) - L(\mathbf{G}(t)) \leq \frac{1}{2}\boldsymbol{\mu}^{\mathsf{T}}(t)\boldsymbol{\mu}(t) + \mathbf{R}^{\mathsf{T}}(t)\mathbf{R}(t) + (\mathbf{R}(t) - \boldsymbol{\mu}(t))^{\mathsf{T}}\mathbf{Q}(t)$$

$$+ \frac{1}{2}(\boldsymbol{\gamma}(t) - \mathbf{D}(t))^{\mathsf{T}}(\boldsymbol{\gamma}(t) - \mathbf{D}(t)) + (\boldsymbol{\gamma}(t) - \mathbf{D}(t))^{\mathsf{T}}\boldsymbol{\Theta}(t) \tag{22}$$

$$\leq B + (\mathbf{R}(t) - \boldsymbol{\mu}(t))^{\mathsf{T}}\mathbf{Q}(t) + (\boldsymbol{\gamma}(t) - \mathbf{D}(t))^{\mathsf{T}}\boldsymbol{\Theta}(t).$$

where B is a uniform bound on the term $\frac{1}{2}\left[\boldsymbol{\mu}^{\mathsf{T}}(t)\boldsymbol{\mu}(t) + \mathbf{R}^{\mathsf{T}}(t)\mathbf{R}(t)\right] + \frac{1}{2}(\boldsymbol{\gamma}(t) - \mathbf{D}(t))^{\mathsf{T}}(\boldsymbol{\gamma}(t) - \mathbf{D}(t))$, that exists under the realistic assumption that the source coding rates, the channel coding rates and the video quality measures are upper bounded by some constants, independent of t. The conditional expected *Lyapunov drift* for slot t is defined by

$$\Delta(\mathbf{G}(t)) := \mathbb{E}\left[L(\mathbf{G}(t+1))|\mathbf{G}(t)\right] - L(\mathbf{G}(t)) \tag{23}$$

Adding on both sides the penalty term $-V\sum_{u\in\mathcal{U}}\mathbb{E}\left[\phi_u(\gamma_u(t))|\mathbf{G}(t)\right]$, we have

$$\Delta(\mathbf{G}(t)) - V\sum_{u\in\mathcal{U}}\mathbb{E}\left[\phi_u(\gamma_u(t))|\mathbf{G}(t)\right] \leq B - V\sum_{u\in\mathcal{U}}\mathbb{E}\left[\phi_u(\gamma_u(t))|\mathbf{G}(t)\right] + \mathbb{E}\left[(\mathbf{R}(t) - \boldsymbol{\mu}(t))^{\mathsf{T}}\mathbf{Q}(t)|\mathbf{G}(t)\right]$$

$$+ \mathbb{E}\left[(\boldsymbol{\gamma}(t) - \mathbf{D}(t))^{\mathsf{T}}\boldsymbol{\Theta}(t)|\mathbf{G}(t)\right]. \tag{24}$$

Here, $V > 0$ is a control parameter of the DPP policy that affects the utility-delay tradeoff. The DPP policy acquires information about $\mathbf{G}(t)$ and $\omega(t)$ at every slot t and chooses $\alpha(t) \in A_{\omega(t)}$, $\gamma_u(t) \, \forall \, u \, \in \, \mathcal{U}$ to minimize the right hand side of the above inequality. Then, the resulting DPP policy is given by the minimization, at each chunk time t, of the function:

$$\underbrace{\mathbf{R}^{\mathsf{T}}(t)\mathbf{Q}(t) - \mathbf{D}^{\mathsf{T}}(t)\boldsymbol{\Theta}(t)}_{\text{admission control}} - \underbrace{\boldsymbol{\mu}^{\mathsf{T}}(t)\mathbf{Q}(t)}_{\substack{\text{transmission} \\ \text{scheduling}}} - \underbrace{\left[V\sum_{u\in\mathcal{U}}\phi_u(\gamma_u(t)) - \boldsymbol{\gamma}^{\mathsf{T}}(t)\boldsymbol{\Theta}(t)\right]}_{\text{obj. maximization}} \tag{25}$$

The choice of $\mathbf{R}(t)$ and of $m_u(t) \, \forall \, u \in \mathcal{U}$ affects only the term $\mathbf{R}^{\mathsf{T}}(t)\mathbf{Q}(t) - \mathbf{D}^{\mathsf{T}}(t)\boldsymbol{\Theta}(t)$, the choice of $\boldsymbol{\mu}(t)$ affects only the term $\boldsymbol{\mu}^{\mathsf{T}}(t)\mathbf{Q}(t)$, and the choice of $\boldsymbol{\gamma}(t)$ affects only the term $V\sum_{u\in\mathcal{U}}\phi_u(\gamma_u(t)) - \boldsymbol{\gamma}^{\mathsf{T}}(t)\boldsymbol{\Theta}(t)$. Thus, the overall minimization decomposes into three separate sub-problems. The first two sub-problems have a clear operational meaning, and will be referred to as *admission control* and *transmission scheduling*. Admission control consists of choosing $m_u(t)$ (quality indices for chunk requested at time t by user u) and the rates $R_{hu}(t)$ of source-coded bits requested by each user u from its

neighboring helpers $h \in \mathcal{N}(u)$. Transmission scheduling consists of allocating the channel transmission rates $\mu_{hu}(t)$ for each helper h to its neighboring users $u \in \mathcal{N}(h)$. The third sub-problem involves the greedy maximization of each user network utility function with respect to the auxiliary control variables $\gamma_u(t)$.

A. Admission Control

The admission control sub-problem objective function (see (25)) can be explicitly expressed as

$$\sum_{u \in \mathcal{U}} \left\{ \sum_{h \in \mathcal{N}(u) \cap \mathcal{H}(f_u)} k Q_{hu}(t) R_{hu}(t) - \Theta_u(t) D_{f_u}(m_u(t), t) \right\}. \tag{26}$$

The minimization of this quantity decomposes into separate minimizations for each user, namely, for each $u \in \mathcal{U}$, choose $m_u(t)$ and $R_{hu}(t) \ \forall \ h \in \mathcal{N}(u) \cap \mathcal{H}(f_u)$ to minimize

$$\sum_{h \in \mathcal{N}(u) \cap \mathcal{H}(f_u)} k Q_{hu}(t) R_{hu}(t) - \Theta_u(t) D_{f_u}(m_u(t), t) \tag{27}$$

with $\{R_{hu}(t)\}_{h \in \mathcal{N}(u) \cap \mathcal{H}(f_u)}$ satisfying (1). It is immediate to see that the solution consists of choosing the helper $h_u^* \in \mathcal{N}(u) \cap \mathcal{H}(f_u)$ with the smallest queue backlog $Q_{hu}(t)$, and assigning the entire requested chunk to h_u^*. Notice that in this way the streaming of the video file f_u may be handled by *different* helpers across the streaming session, but each individual chunk is downloaded from a single helper. Further, the quality mode $m_u(t)$ is chosen as

$$m_u(t) = \arg \min_{m \in \{1, \ldots, N_{f_u}\}} \left\{ k Q_{h_u^* u}(t) B_{f_u}(m, t) - \Theta_u(t) D_{f_u}(m, t) \right\}. \tag{28}$$

In order to implement this policy, it is sufficient that each user knows only its *local information* of the queue backlogs of its neighboring helpers. Congestion control decisions are decentralized: each user u chooses the helper $h \in \mathcal{N}(u) \cap \mathcal{H}(f_u)$ with minimum backlog, and requests the t-th chunk at a quality level according to (28). This policy is reminiscent of the current adaptive streaming technology for video on demand systems, referred to as DASH (Dynamic Adaptive Streaming over HTTP) [15], [16], where the client (user) progressively fetches a video file by downloading successive chunks, and makes adaptive decisions on the source encoding quality based on its current knowledge of the congestion of the underlying server-client connection.

B. Transmission Scheduling

Transmission scheduling involves maximizing the weighted sum rate $\sum_{h \in \mathcal{H}} \sum_{u \in \mathcal{N}(h)} Q_{hu}(t) \mu_{hu}(t)$ where the weights are the queue backlogs (see (25)). Under our system assumptions, this problem

decouples into separate maximizations for each helper. Notice that here, unlike conventional cellular systems, we do not assign a fixed set of users to each helper. In contrast, the helper-user association is dynamic, and results from the transmission scheduling decision itself. For each $h \in \mathcal{H}$, the transmission scheduling problem can be written as the *Linear Program* (LP):

$$\text{maximize} \quad \sum_{u \in \mathcal{N}(h)} Q_{hu}(t)\mu_{hu}(t) \tag{29}$$

$$\text{subject to} \quad \sum_{u \in \mathcal{N}(h)} \frac{\mu_{hu}(t)}{C_{hu}(t)} \leq 1. \tag{30}$$

By a change of variables $\nu_{hu}(t) = \frac{\mu_{hu}(t)}{C_{hu}(t)}$, the LP reduces to

$$\text{maximize} \quad \sum_{u \in \mathcal{N}(h)} Q_{hu}(t)C_{hu}(t)\nu_{hu}(t) \tag{31}$$

$$\text{subject to} \quad \sum_{u \in \mathcal{N}(h)} \nu_{hu}(t) \leq 1. \tag{32}$$

The feasible region of the above LP is the $|\mathcal{N}(h)|$-simplex polytope and it is immediate to see that the solution consists of scheduling the user $u_h^* \in \mathcal{N}(h)$ with the largest product $Q_{hu}(t)C_{hu}(t)$, and serve this user at rate $\mu_{hu_h^*}(t) = C_{hu_h^*}(t)$, while all other queues of helper h are not served in slot t.

C. Greedy maximization of the network utility function

Each user $u \in \mathcal{U}$ keeps track of $\Theta_u(t)$ and chooses its virtual queue arrival $\gamma_u(t)$ in order to solve:

$$\text{maximize} \quad V\phi_u(\gamma_u(t)) - \Theta_u(t)\gamma_u(t) \tag{33}$$

$$\text{subject to} \quad D_u^{\min} \leq \gamma_u(t) \leq D_u^{\max}. \tag{34}$$

These decisions push the system to approach the maximum of the network utility function. By appropriately choosing the functions ϕ_u, we can impose some desired notion of fairness. For example, a general class of concave functions suitable for this purpose is given by the α-fairness network utility, defined by [25]

$$\phi_u(x) = \varphi_\alpha(x) = \begin{cases} \log x & \alpha = 1 \\ \frac{x^{1-\alpha}}{1-\alpha} & \alpha > 0, \ \alpha \neq 1 \end{cases} \tag{35}$$

In this case, it is well-known that $\alpha = 0$ yields the maximization of the sum quality (no fairness), $\alpha \to \infty$ yields the maximization of the worst-case quality (max-min fairness) and $\alpha = 1$ yields the maximization of the geometric mean quality (proportional fairness).

IV. Performance Analysis: Arbitrarily Time Varying Network State

As outlined in Section II, VBR video encoding yields a time-varying quality-rate tradeoff that depends on the individual video file, and arbitrary user motion yields time variations of the path coefficients at the time-scale of the video streaming session. As a result, any stationarity or ergodicity assumption about the network state process $\omega(t)$ is unlikely to hold in most practically relevant settings. Therefore, in this section we consider performance guarantees of the DPP policy for *any arbitrary sample path* $\omega(t)$. We adopt the *universal scheduling* technique developed in [3] and [18] to address this scenario. We use a metric called the *T-slot lookahead metric* for evaluating the DPP policy. Time is split into frames of duration T slots and we consider F such frames. For any arbitrary sample path $\omega(t)$, we consider the static optimization problem over the j-th frame:

$$\max \quad \phi_j := \sum_{u \in \mathcal{U}} \phi_u \left(\frac{1}{T} \sum_{\tau=jT}^{(j+1)T-1} D_u(\tau) \right) \tag{36}$$

$$\text{subject to} \quad \frac{1}{T} \sum_{\tau=jT}^{(j+1)T-1} [kR_{hu}(\tau) - n\mu_{hu}(\tau)] \leq 0 \ \forall \ (h,u) \in \mathcal{E} \tag{37}$$

$$\alpha(t) \in A_{\omega(t)} \ \forall \ t \ \in \ \{jT, \ldots, (j+1)T-1\} \tag{38}$$

We denote by ϕ_j^* the optimal solution to the above problem. Thus, the value ϕ_j^* is the maximum utility for frame j that can be achieved over all policies which have future knowledge of the sample path $\omega(t)$ over the j-th frame, subject to the constraint (37) which ensures that for every queue Q_{hu}, the total service provided over the frame is at least as large as the total arrivals in that frame. As in Section III, we consider the following equivalent problem which involves only time averages and introduces the auxiliary variables $\gamma_u(t)$:

$$\max \quad \frac{1}{T} \sum_{\tau=jT}^{(j+1)T-1} \sum_{u \in \mathcal{U}} \phi_u(\gamma_u(\tau)) \tag{39}$$

$$\text{subject to} \quad \frac{1}{T} \sum_{\tau=jT}^{(j+1)T-1} [kR_{hu}(\tau) - n\mu_{hu}(\tau)] \leq 0 \ \forall \ (h,u) \in \mathcal{E} \tag{40}$$

$$\frac{1}{T} \sum_{\tau=jT}^{(j+1)T-1} [\gamma_u(\tau) - D_u(\tau)] \leq 0 \ \forall \ u \in \mathcal{U} \tag{41}$$

$$D_u^{\min} \leq \gamma_u(t) \leq D_u^{\max} \ \forall \ u \in \mathcal{U}, \ \forall \ t \in \{jT, \ldots, (j+1)T-1\} \tag{42}$$

$$\alpha(t) \in A_{\omega(t)} \ \forall \ t \ \in \ \{jT, \ldots, (j+1)T-1\}. \tag{43}$$

The update equations for the transmission queues $Q_{hu} \ \forall \ (h, u) \in \mathcal{E}$ and the virtual queues $\Theta_u \ \forall \ u \in \mathcal{U}$ remain the same as (2) and (9) respectively. Let $\mathbf{G}(t) = \left[\mathbf{Q}^{\mathsf{T}}(t), \mathbf{\Theta}^{\mathsf{T}}(t) \right]^{\mathsf{T}}$ be the combined queue backlogs column vector, and define the quadratic Lyapunov function

$$L(\mathbf{G}(t)) = \frac{1}{2} \mathbf{G}^{\mathsf{T}}(t) \mathbf{G}(t).$$

Fix a particular slot τ in the j-th frame. We first consider the 1-slot drift of $L(\mathbf{G}(\tau))$. From (22), we know that

$$L(\mathbf{G}(\tau + 1)) - L(\mathbf{G}(\tau)) \leq B + (\mathbf{R}(t) - \boldsymbol{\mu}(\tau))^{\mathsf{T}} \mathbf{Q}(\tau) + (\boldsymbol{\gamma}(\tau) - \mathbf{D}(t))^{\mathsf{T}} \mathbf{\Theta}(\tau) \qquad (44)$$

where B is a uniform bound on the term $\frac{1}{2} \left[\boldsymbol{\mu}^{\mathsf{T}}(t) \boldsymbol{\mu}(t) + \mathbf{R}^{\mathsf{T}}(t) \mathbf{R}(t) \right] + \frac{1}{2} \left(\boldsymbol{\gamma}(t) - \mathbf{D}(t) \right)^{\mathsf{T}} \left(\boldsymbol{\gamma}(t) - \mathbf{D}(t) \right)$, that exists under the realistic assumption that the source coding rates, the channel coding rates and the video quality measures are upper bounded by some constants, independent of t. We choose B such that

$$B > 2 \boldsymbol{\kappa}^{\mathsf{T}} \boldsymbol{\kappa} \qquad (45)$$

where $\boldsymbol{\kappa}$ is a vector whose components are all equal to the same number κ and this number is a uniform upper bound on the maximum possible magnitude of drift in any of the queues (both actual and virtual) in one slot.

With the additional penalty term $-V \sum_{u \in \mathcal{U}} \phi_u(\gamma_u(\tau))$ added on both sides of (44), we have the following *drift plus penalty* inequality:

$$L(\mathbf{G}(\tau + 1)) - L(\mathbf{G}(\tau)) - V \sum_{u \in \mathcal{U}} \phi_u(\gamma_u(\tau)) \leq B + (\mathbf{R}(t) - \boldsymbol{\mu}(\tau))^{\mathsf{T}} \mathbf{Q}(\tau) + (\boldsymbol{\gamma}(\tau) - \mathbf{D}(t))^{\mathsf{T}} \mathbf{\Theta}(\tau)$$

$$- V \sum_{u \in \mathcal{U}} \phi_u(\gamma_u(\tau)) \qquad (46)$$

Let $\{\alpha(\tau)\}_{\tau = jT}^{(j+1)T-1}$ denote the DPP policy which minimizes the right hand side of the *drift plus penalty* inequality (46). Since it minimizes the expression on the RHS of (46), any other policy $\{\alpha^*(\tau)\}_{\tau = jT}^{(j+1)T-1}$ comprising of the decisions $\{m_u^*(\tau)\}_{\tau = jT}^{(j+1)T-1}$, $\{\mathbf{R}^*(\tau)\}_{\tau = jT}^{(j+1)T-1}$, $\{\boldsymbol{\mu}^*(\tau)\}_{\tau = jT}^{(j+1)T-1}$ and $\{\boldsymbol{\gamma}^*(\tau)\}_{\tau = jT}^{(j+1)T-1}$ would give a larger value of the expression. We therefore have

$$L(\mathbf{G}(\tau + 1)) - L(\mathbf{G}(\tau)) - V \sum_{u \in \mathcal{U}} \phi_u(\gamma_u(\tau)) \leq B + (\mathbf{R}^*(\tau) - \boldsymbol{\mu}^*(\tau))^{\mathsf{T}} \mathbf{Q}(\tau) + (\boldsymbol{\gamma}^*(\tau) - \mathbf{D}^*(\tau))^{\mathsf{T}} \mathbf{\Theta}(\tau)$$

$$- V \sum_{u \in \mathcal{U}} \phi_u(\gamma_u^*(\tau)). \qquad (47)$$

Further, we note that the maximum change in the queue length vectors $Q_{hu}(\tau)$ and $\Theta_u(\tau)$ from one slot to the next is bounded by κ. Thus, we have for all $\tau \in \{jT, \ldots, (j+1)T-1\}$

$$|Q_{hu}(\tau) - Q_{hu}(jT)| \leq (\tau - jT)\kappa \quad \forall \, (h,u) \in \mathcal{E} \tag{48}$$

$$|\Theta_u(\tau) - \Theta_u(jT)| \leq (\tau - jT)\kappa \quad \forall \, u \in \mathcal{U} \tag{49}$$

Substituting the above inequalities in (47), we have

$$L(\mathbf{G}(\tau+1)) - L(\mathbf{G}(\tau)) - V \sum_{u \in \mathcal{U}} \phi_u(\gamma_u(\tau)) \leq B + (\mathbf{R}^*(\tau) - \boldsymbol{\mu}^*(\tau))^\mathsf{T} (\mathbf{Q}(jT) + (\tau - jT)\boldsymbol{\kappa})$$

$$+ (\boldsymbol{\gamma}^*(\tau) - \mathbf{D}^*(\tau))^\mathsf{T} (\boldsymbol{\Theta}(jT) + (\tau - jT)\boldsymbol{\kappa})$$

$$- V \sum_{u \in \mathcal{U}} \phi_u(\gamma_u^*(\tau)). \tag{50}$$

Then, summing (50) over $\tau \in \{jT, \ldots, (j+1)T-1\}$, we obtain the T-slot Lyapunov drift over the j-th frame:

$$L(\mathbf{G}((j+1)T)) - L(\mathbf{G}(jT)) - V \sum_{\tau=jT}^{jT+T-1} \sum_{u \in \mathcal{U}} \phi_u(\gamma_u(\tau))$$

$$\leq BT + \left(\sum_{\tau=jT}^{jT+T-1} (\mathbf{R}^*(\tau) - \boldsymbol{\mu}^*(\tau)) \right)^\mathsf{T} \mathbf{Q}(jT) + \left(\sum_{\tau=jT}^{jT+T-1} (\mathbf{R}^*(\tau) - \boldsymbol{\mu}^*(\tau))(\tau - jT) \right)^\mathsf{T} \boldsymbol{\kappa}$$

$$+ \left(\sum_{\tau=jT}^{jT+T-1} (\boldsymbol{\gamma}^*(\tau) - \mathbf{D}^*(\tau)) \right)^\mathsf{T} \boldsymbol{\Theta}(jT) + \left(\sum_{\tau=jT}^{jT+T-1} (\boldsymbol{\gamma}^*(\tau) - \mathbf{D}^*(\tau))(\tau - jT) \right)^\mathsf{T} \boldsymbol{\kappa}$$

$$- V \sum_{\tau=jT}^{jT+T-1} \sum_{u \in \mathcal{U}} \phi_u(\gamma_u^*(\tau)) \tag{51}$$

Using the inequalities $\mathbf{R}^*(\tau) - \boldsymbol{\mu}^*(\tau) \leq 2\boldsymbol{\kappa}$, $\boldsymbol{\gamma}^*(\tau) - \mathbf{D}^*(\tau) \leq 2\boldsymbol{\kappa}$ in (51), we have

$$L(\mathbf{G}((j+1)T)) - L(\mathbf{G}(jT)) - V \sum_{\tau=jT}^{jT+T-1} \sum_{u \in \mathcal{U}} \phi_u(\gamma_u(\tau))$$

$$\leq BT + \left(\sum_{\tau=jT}^{jT+T-1} (\mathbf{R}^*(\tau) - \boldsymbol{\mu}^*(\tau)) \right)^\mathsf{T} \mathbf{Q}(jT) + 2 \left(\sum_{\tau=jT}^{jT+T-1} (\tau - jT) \right) \boldsymbol{\kappa}^\mathsf{T} \boldsymbol{\kappa}$$

$$+ \left(\sum_{\tau=jT}^{jT+T-1} (\boldsymbol{\gamma}^*(\tau) - \mathbf{D}^*(\tau)) \right)^\mathsf{T} \boldsymbol{\Theta}(jT) + 2 \left(\sum_{\tau=jT}^{jT+T-1} (\tau - jT) \right) \boldsymbol{\kappa}^\mathsf{T} \boldsymbol{\kappa}$$

$$- V \sum_{\tau=jT}^{jT+T-1} \sum_{u \in \mathcal{U}} \phi_u(\gamma_u^*(\tau)) \tag{52}$$

Using $\boldsymbol{\kappa}^{\mathsf{T}}\boldsymbol{\kappa} \leq \frac{B}{2}$, $\sum_{\tau=jT}^{jT+T-1}(\tau - jT) = \frac{T(T-1)}{2}$, we get

$$
L(\mathbf{G}((j+1)T)) - L(\mathbf{G}(jT)) - V \sum_{\tau=jT}^{jT+T-1} \sum_{u \in \mathcal{U}} \phi_u(\gamma_u(\tau))
$$

$$
\leq BT + BT(T-1) + \left(\sum_{\tau=jT}^{jT+T-1} (\mathbf{R}^*(\tau) - \boldsymbol{\mu}^*(\tau)) \right)^{\mathsf{T}} \mathbf{Q}(jT)
$$

$$
+ \left(\sum_{\tau=jT}^{jT+T-1} (\boldsymbol{\gamma}^*(\tau) - \mathbf{D}^*(\tau)) \right)^{\mathsf{T}} \boldsymbol{\Theta}(jT) - V \sum_{\tau=jT}^{jT+T-1} \sum_{u \in \mathcal{U}} \phi_u(\gamma_u^*(\tau)) \tag{53}
$$

We now consider the policy $\{\alpha^*(\tau)\}_{\tau=jT}^{(j+1)T-1}$ satisfying the following constraints:

$$
\frac{1}{T} \sum_{\tau=jT}^{(j+1)T-1} [kR_{hu}^*(\tau) - n\mu_{hu}^*(\tau)] < -\epsilon \ \forall \ (h, u) \in \mathcal{E} \tag{54}
$$

$$
\frac{1}{T} \sum_{\tau=jT}^{(j+1)T-1} [\gamma_u^*(\tau) - D_u^*(\tau)] < -\epsilon \ \forall \ u \in \mathcal{U} \tag{55}
$$

where $\epsilon > 0$ is arbitrary. We plug in the inequalities (54), (55) in (53) and obtain

$$
L(\mathbf{G}((j+1)T)) - L(\mathbf{G}(jT)) - V \sum_{\tau=jT}^{jT+T-1} \sum_{u \in \mathcal{U}} \phi_u(\gamma_u(\tau))
$$

$$
< BT^2 - \epsilon T \sum_{(h,u) \in \mathcal{E}} Q_{hu}(jT) - \epsilon T \sum_{u \in \mathcal{U}} \Theta_u(jT) - V \sum_{\tau=jT}^{jT+T-1} \sum_{u \in \mathcal{U}} \phi_u(\gamma_u^*(\tau)) \tag{56}
$$

Also, considering the fact that for any vector $\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_{|\mathcal{U}|})$ we have

$$
\sum_{u \in \mathcal{U}} \phi_u(D_u^{\min}) = \phi_{\min} \leq \sum_{u \in \mathcal{U}} \phi_u(\gamma_u) \leq \phi_{\max} = \sum_{u \in \mathcal{U}} \phi_u(D_u^{\max}), \tag{57}
$$

we can write:

$$
L(\mathbf{G}((j+1)T)) - L(\mathbf{G}(jT)) < BT^2 + VT(\phi_{\max} - \phi_{\min}) - \epsilon T \sum_{(h,u) \in \mathcal{E}} Q_{hu}(jT) - \epsilon T \sum_{u \in \mathcal{U}} \Theta_u(jT)
$$

$$
\tag{58}
$$

Once again using (48), (49), we have:

$$
L(\mathbf{G}((j+1)T)) - L(\mathbf{G}(jT)) < BT^2 + VT(\phi_{\max} - \phi_{\min}) - \epsilon \sum_{\tau=jT}^{jT+T-1} \sum_{(h,u) \in \mathcal{E}} Q_{hu}(\tau)
$$

$$
- \epsilon \sum_{\tau=jT}^{jT+T-1} \sum_{u \in \mathcal{U}} \Theta_u(\tau) + \frac{\epsilon \kappa(|\mathcal{E}| + |\mathcal{U}|)T(T-1)}{2} \tag{59}
$$

Summing the above over the frames $j \in \{0, \ldots, F-1\}$ yields

$$L(\mathbf{G}((FT)) - L(\mathbf{G}(0)) < BT^2 F + VFT(\phi_{\max} - \phi_{\min}) - \epsilon \sum_{\tau=0}^{FT-1} \sum_{(h,u)\in\mathcal{E}} Q_{hu}(\tau)$$

$$- \epsilon \sum_{\tau=0}^{FT-1} \sum_{u\in\mathcal{U}} \Theta_u(\tau) + \frac{\epsilon\kappa(|\mathcal{E}| + |\mathcal{U}|)FT(T-1)}{2} \quad (60)$$

Rearranging and neglecting appropriate terms, we get

$$\frac{1}{FT} \sum_{\tau=0}^{FT-1} \sum_{(h,u)\in\mathcal{E}} Q_{hu}(\tau) + \frac{1}{FT} \sum_{\tau=0}^{FT-1} \sum_{u\in\mathcal{U}} \Theta_u(\tau) < \frac{BT}{\epsilon} + \frac{V(\phi_{\max} - \phi_{\min})}{\epsilon} + \frac{L(\mathbf{G}(0))}{\epsilon FT}$$

$$+ \frac{\kappa(|\mathcal{E}| + |\mathcal{U}|)(T-1)}{2} \quad (61)$$

Taking limits as $F \to \infty$

$$\boxed{\lim_{F\to\infty} \frac{1}{FT} \sum_{\tau=0}^{FT-1} \left(\sum_{(h,u)\in\mathcal{E}} Q_{hu}(\tau) + \sum_{u\in\mathcal{U}} \Theta_u(\tau) \right) < \frac{BT}{\epsilon} + \frac{V(\phi_{\max} - \phi_{\min})}{\epsilon} + \frac{\kappa(|\mathcal{E}| + |\mathcal{U}|)(T-1)}{2}.}$$

$$(62)$$

We now consider the policy $\{\alpha^*(\tau)\}_{\tau=jT}^{(j+1)T-1}$ which achieves the optimal solution ϕ_j^* to the problem (39) – (43). Using (40) and (41) in (53), we have

$$L(\mathbf{G}((j+1)T)) - L(\mathbf{G}(jT)) - V \sum_{\tau=jT}^{jT+T-1} \sum_{u\in\mathcal{U}} \phi_u(\gamma_u(\tau)) \leq BT + BT(T-1) - VT\phi_f^* \quad (63)$$

Summing this over $j \in \{0, \ldots, F-1\}$, yields

$$L(\mathbf{G}((FT)) - L(\mathbf{G}(0)) - V \sum_{\tau=0}^{FT-1} \sum_{u\in\mathcal{U}} \phi_u(\gamma_u(\tau)) \leq BT^2 F - VT \sum_{r=0}^{F-1} \phi_f^*. \quad (64)$$

Dividing both sides by VFT and using the fact that $L(\mathbf{G}((FT)) > 0$, we get

$$\frac{1}{FT} \sum_{\tau=0}^{FT-1} \sum_{u\in\mathcal{U}} \phi_u(\gamma_u(\tau)) \geq \frac{1}{F} \sum_{f=0}^{F-1} \phi_f^* - \frac{BT}{V} - \frac{L(\mathbf{G}(0))}{VTF}. \quad (65)$$

At this point, using Jensen's inequality, the fact that $\phi_u(\cdot)$ is continuous and non-decreasing for all $u \in \mathcal{U}$, and the fact that the strong stability of the queues (62) implies that $\lim_{F\to\infty} \frac{1}{FT} \sum_{\tau=0}^{FT-1} \Theta_u(\tau) < \infty \ \forall \ u \in \mathcal{U}$, which in turns implies that $\overline{\gamma}_u \leq \overline{D}_u \ \forall \ u \in \mathcal{U}$, we arrive at

$$\boxed{\sum_{u\in\mathcal{U}} \phi_u\left(\overline{D}_u\right) \geq \lim_{F\to\infty} \frac{1}{F} \sum_{f=0}^{F-1} \phi_f^* - \frac{BT}{V}.} \quad (66)$$

Thus, the utility under the DPP policy is within $O(1/V)$ of the time average of the ϕ_j^* values which can be achieved only if knowledge of the future states up to a look-ahead of blocks of T slots. If T is

increased, then the value of ϕ_j^* for every frame j improves since we allow a larger look-ahead. However, from (66), we can see that if T is increased, then V also has to be increased to maintain the same distance from optimality which in turn would prompt an $O(V)$ increase in the queue backlog. Similar performance guarantees for the case when $\omega(t)$ is i.i.d can be recovered as a special case of the above analysis and are omitted.

V. PRE-BUFFERING, RE-BUFFERING AND SKIPPING CHUNKS

As described in Section I, the playback process consumes chunks at fixed playback rate (one chunk per time slot) while the number of *ordered* chunks per unit time entering the playback buffer is a random variable, due to the fact that the transmission resources are dynamically allocated by the scheduling policy, and that the network state $\omega(t)$ may be time-varying. In addition, it may happen that chunks which go through different queues in the network are affected by different delays. This may give rise to a situation where already received chunks with higher order number cannot be used for playback until the missing chunks with lower order number are also received. An example is illustrated in Table I and Figure 1. The table indicates the chunk numbers and their respective arrival times. The blue curve in Figure 1 shows the evolution with time of the number of ordered chunks which are available and are ready for playback. On the other hand, the green curve indicates the evolution with time of the number of chunks consumed in playback. The playback consumption starts after an initial pre-buffering or startup delay d as indicated in the figure. At any instant t, the chunk requested at $t - d$ is expected to be available in the playback buffer. However, if the chunk is delivered with a delay greater than d, the two curves meet and there is an interruption/stall event in the playback process. Hence, in order to prevent the stall event, a user would ideally like to choose d to be larger than the maximum delay in delivering the chunks. Unfortunately, such maximum delay is neither deterministic nor known a priori. In fact, it depends on the dynamics of the network state $\omega(t)$ (including user mobility, streaming activity and the non-ergodic non-stationary dynamics of the video VBR coding, as discussed in Section I). Hence, this maximum delay must be estimated adaptively and then the initial pre-buffering time and the re-buffering time in the case of a stall event must be chosen such that the number of ordered chunks in the buffer is sufficiently larger than the observed maximum delay. Every user u can estimate its local delays by monitoring its requests and delivery time in a sliding window spanning a fixed number of time slots in the past. Finally, a user can also skip a chunk if by doing so, it can provide a large jump-up in the number of ordered chunks. For instance, in Table I and Figure 1, the chunk which comes 4th in the ordered sequence arrives at the end of time slot 11. However, chunks numbered $5, 6, 7$ and 8 arrive before slot 11 but cannot be played since

TABLE I: Arrival times of chunks

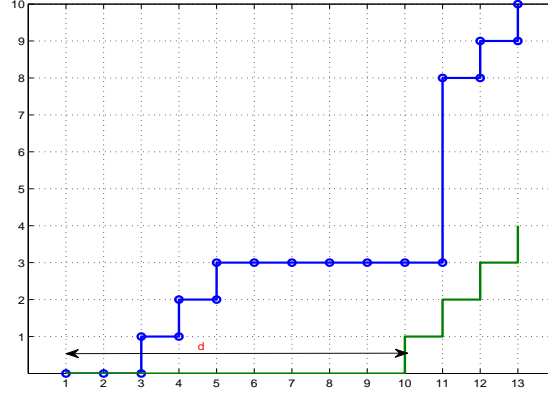| 1	2	3	4	5	6	7	8	9	10	11	12	13
3	4	5	11	6	8	9	10	12	13	16	15	14

Fig. 1: Evolution of number of ordered and consumed chunks

4 is missing. More generally, if chunk 4 were to arrive with a delay such that the number of chunks which arrive before 4 but come later in the ordered sequence becomes large, then the user could either continue waiting for the missing chunk and incur a stall event or it could be worthwhile to skip chunk 4 from playback and provide a large jump-up in the buffer.

We now describe formally the proposed policy for pre-buffering, re-buffering and chunk skipping that each user u implements locally. Let T_k denote the time slot in which a user requests the kth chunk and let A_k be the time slot in which the chunk arrives at the user playback buffer. The delay for chunk k is $W_k = A_k - T_k$. Without loss of generality, consider a user u starting its streaming session at time $t = 1$. In the proposed scheduling policy, user u requests one chunk per scheduling time, sequentially and possibly from different helpers, such that $T_1 = 1$, $T_2 = 2$, and so on. Since the chunks are downloaded from different neighboring helpers with different queue lengths and downlink channel capacities, they may be received out of order. For example, it may happen that $A_k < A_j$ for some $j < k$. Hence, chunk k cannot be played until all chunks j for $j < k$ are also received. We say that a chunk k becomes *playable* when all the chunks $j \leq k$ are received. Let P_k denote the time when chunk k becomes *playable*. Then, we have:

$$P_k = \max\{A_1, A_2, \ldots, A_k\}. \tag{67}$$

The proposed policy consists of two parts: skipping chunks from playback and buffering policy. We examine these two features separately in the following.

A. Skipping chunks from playback

Prior to slot t, the set of playable chunks is $\{k : P_k \leq t - 1\}$ and

$$k_{t-1}^* = \max\{k : P_k \leq t - 1\} \tag{68}$$

is the highest-order chunk in the ordered sequence of playable chunks. At the end of slot t, user u considers the set C_t of all chunks which have arrived before or during slot t and which come later than k_{t-1}^* in the ordered sequence of playback. The set C_t is given by:

$$C_t = \{k : k > k_{t-1}^*, \ A_k \leq t\}. \tag{69}$$

The next available chunk with order larger than k_{t-1}^* is given by:

$$k_t^- = \min\{k : k > k_{t-1}^*, \ A_k \leq t\}. \tag{70}$$

Let $C_t^* \subseteq C_t$ be the set of chunks which become playable at the end of slot t, i.e.,

$$C_t^* = \{k : A_k \leq t, P_k = t\}. \tag{71}$$

If k_t^- comes next to k_{t-1}^* in the playback order (i.e. if $k_t^- = k_{t-1}^* + 1$), then C_t^* is non-empty and all the chunks $k \in C_t^*$ can be added to the playback buffer. Further, k_t^* is recursively updated as:

$$k_t^* = k_{t-1}^* + |C_t^*|. \tag{72}$$

Denoting the increment in the size of the playback buffer at the end of slot t by Λ_t, we have in this case that $\Lambda_t = |C_t^*|$. On the other hand, if k_t^- is not the immediate successor of k_{t-1}^* in the playback order (i.e. if $k_t^- > k_{t-1}^* + 1$), then there is no chunk in C_t which becomes playable at the end of slot t and therefore $C_t^* = \emptyset$. In this case, the algorithm compares $|C_t|$ with a threshold ρ in order to decide whether it should wait further for the missing chunk $k_{t-1}^* + 1$ or skip it in order to increase the playback buffer anyway. As explained in the qualitative discussion at the beginning of Section V, the intuition behind such a decision is that it is worthwhile to skip a chunk if skipping such a chunk results in a large jump in the playback buffer size. The size of this possible jump can be exactly computed from C_t as follows: assuming $k_t^- = k_{t-1}^* + 2$, if we skip chunk $k_{t-1}^* + 1$, then the increase in the playback buffer is given by the size of the set:

$$\{j \geq 2 \ : \ k_{t-1}^* + i \in C_t \ \forall \ 2 \leq i \leq j\}.$$

We therefore propose the following policy: if $|C_t| \leq \rho$ (where ρ is a parameter > 0), then wait for chunk $k^*_{t-1}+1$ and let $k^*_t = k^*_{t-1}$. Otherwise, if $|C_t| > \rho$, the increase of the playback buffer is worthwhile and therefore it is useful to skip chunk $k^*_{t-1} + 1$. In this case, if $k^-_t = k^*_{t-1} + 2$, then k^*_t is updated as

$$k^*_t = k^*_{t-1} + \max\{j : k^*_{t-1} + i \ \forall \ 2 \leq i \leq j\} \tag{73}$$

and all the chunks numbered from $k^*_{t-1} + 2$ to k^*_t are made playable at the end of slot t and added to the playback buffer. We therefore have $\Lambda_t = |\{j > 2 : k^*_{t-1} + i \ \forall \ 2 \leq i \leq j\}|$ in this case. Instead, if $k^-_t > k^*_{t-1} + 2$, then the user skips chunk $k^*_{t-1} + 1$ and starts waiting for chunk $k^*_{t-1} + 2$. Only a single chunk is allowed to be skipped per slot because skipping multiple chunks might cause damage to the quality of experience of the user. Note that in this case, k^*_t is updated to $k^*_{t-1} + 1$ even though the chunk $k^*_{t-1} + 1$ is missing and is not playable. This is to ensure that when chunk $k^*_{t-1} + 2$ is received, it is considered playable despite the fact that chunk $k^*_{t-1} + 1$ is missing. Also note that there is no increment in the playback buffer (i.e., $\Lambda_t = 0$) in this case because there is no new chunk which becomes playable. Note that choosing $\rho = \infty$ corresponds to the case when no chunk is skipped.

B. Pre-buffering and re-buffering

The goal here is to determine the delay T_u after which user u should start playback, with respect to the time at which the first chunk is requested (beginning of the streaming session). Intuitively, choosing a large T_u makes the probability of stall events small. However, a too large T_u is very annoying for the user's quality of experience. From the chunk skipping strategy seen above, we know that Λ_t is the number of new chunks added to the playback buffer at the end of slot t. We define the size of the playback buffer Ψ_t as the number of playable chunks in the buffer not yet played. Without loss of generality, assume again that the streaming session starts at $t = 1$. Then, Ψ_t is recursively given by the updating equation:[5]

$$\Psi_t = \max\{\Psi_{t-1} - 1\{t > T_u\}, 0\} + \Lambda_t. \tag{74}$$

From the qualitative discussion on the evolution of the playback buffer at the beginning of Section V, we notice that the longest period during which Ψ_t is not incremented (in the absence of chunk skipping decisions) is given by the maximum delay W_k to deliver chunks. In addition, we note that each user u needs to adaptively estimate W_k in order to choose T_u. In the proposed method, user u calculates for every chunk k the corresponding delay $W_k = A_k - T_k$. Notice that the delay of chunk k, can be calculated only at time A_k, i.e., when the chunk is actually delivered. At each time $t = 1, 2, \ldots$, user u calculates

---

[5]$1\{\mathcal{K}\}$ denotes the indicator function of a condition or event $\mathcal{K}$.

the maximum observed delay E_t in a sliding window of size Δ, (in all the numerical experiments in the sequel, we use $\Delta = 10$) by letting:

$$E_t = \max\{W_k \ : \ t - \Delta + 1 \leq A_k \leq t\}. \tag{75}$$

Finally, user u starts its playback when Ψ_t crosses the level ξE_t, i.e.,

$$T_u = \min\{t : \ \Psi_t \geq \xi E_t\}. \tag{76}$$

If a stall event occurs at time t, i.e., $\Psi_t = 0$ for $t > T_u$, the algorithm enters a re-buffering phase in which the same algorithm presented above is employed again to determine the new instant $t + T_u + 1$ at which playback is restarted. Notice that even if here, with some abuse of notation, we have denoted the re-buffering delay again by T_u, in fact this is re-estimated using the sliding window method at every new stall event. In fact, if a stall event occurs it is likely that some change in the network state has occurred, such that the maximum delay must be re-estimated.

VI. NUMERICAL EXPERIMENTS

We consider a 400m ×400m square area divided into 5×5 small square cells of side length 80m as shown in Figure 2. A helper is located at the center of each small square cell. Each helper serves only those users within a radius of 60 m. As described in Section I, the helpers could be connected to some video content delivery network through a wired backbone or they could be dedicated nodes with local caching capacity. In these simulations we assume that each helper has available the whole video library. Therefore, for any request f_u we have $\mathcal{N}(u) \cap \mathcal{H}(f_u) = \mathcal{N}(u)$. We further assume that there are 2 users uniformly and independently distributed in each small cell. We use the utility function $\phi_u(x) = \log(x)$ for all $u \in \mathcal{U}$. We assume a physical layer inspired by LTE specifications [26]. As described in Section II, with a scheduling slot duration of 0.5s and a total available system bandwidth of 18 MHz, we have 10^5 LTE resource blocks. Thus, the total number of channel symbols n available in a scheduling slot is $10^5 \times 84$.

Between any two points a and b in the square area, the path loss is given by $g(a, b) = \frac{1}{1+\left(\frac{d(a,b)}{\delta}\right)^\alpha}$ where $\delta = 40$ m and $\alpha = 3.5$. Each helper transmits at a power level such that the SNR per symbol (without interference) at the center (i.e., at distance $d(a, b) = 0$ from the transmitter) is 20 dB. The point-to-point ergodic capacities in (3) are calculated by using the approximate formula: $C_{hu}(t) = e^{1/\Gamma_{hu}(t)} \text{Ei}\left(1, \frac{1}{\Gamma_{hu}(t)}\right)$ where $\text{Ei}(1, x) = \int_x^\infty \frac{e^{-t}}{t} dt$ for $x \geq 0$, and $\Gamma_{hu}(t) = \frac{P_h g_{hu}(t)}{1+\sum_{h' \neq h} P_{h'} g_{h'u}(t)}$. This formula is easy to derive and is known to be accurate.
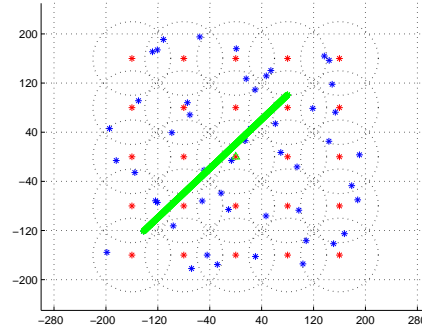
Fig. 2: Toplogy (the green line indicates the trajectory of a mobile user in Experiment 2).

We assume that all the users request chunks successively from VBR-encoded video sequences. Each video file is a long sequence of chunks, each of duration 0.5 seconds and with a frame rate of 30 frames per second. We consider a specific video sequence formed by 800 chunks, constructed using 4 video clips from the database in [27], each of length 200 chunks. The chunks are encoded into different quality modes. Here, the quality index is measured using the *Structural SIMilarity* (SSIM) index defined in [28], Figures 3a and 3b show the size in kbits and the SSIM values as a function of the chunk index, respectively, for the different quality modes. In our experiments, the chunks from 1 to 200 and 601 to 800 are encoded into 8 quality modes, while the chunks numbered from 201 to 600 are encoded in 4 quality modes. In all the numerical experiments in the sequel, each user starts its streaming session of 1000 chunks from some arbitrary position in this reference video sequence and successively requests 1000 chunks by cycling through the sequence.

A. Experiment 1

At $t = 0$, all users start their streaming session as explained before, for the duration of 1000 chunks. We identify the following five key performance metrics: 1) The percentage of skipped chunks due to large delays; 2) The initial pre-buffering time (in number of slots) or the startup delay T_u before starting playback; 3) The number of times playback gets interrupted and enters *re-buffering* mode (frequency of stall events per streaming session); 4) The percentage of total playback time which is spent in *re-buffering* mode; 5) The quality (SSIM) averaged over the delivered (i.e., the chunks which are not skipped) chunks.

The parameters governing the DPP policy and skipping/buffering policy are V, ξ and ρ. We have studied the effect of each individual parameter while keeping the other two fixed to some reasonable value, in terms of the above key performance metrics. Figure 4a shows the cumulative distribution function (CDF)

(a) bitrate profile
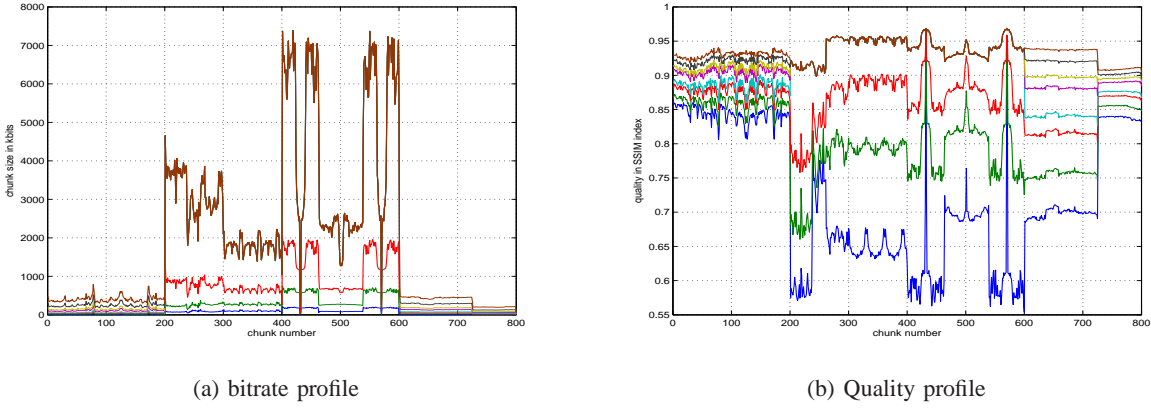
(b) Quality profile

Fig. 3: Rate-quality profile of video sequence.

of quality (averaged over delivered chunks) over the user population for the values $2V, 4V, 6V, 8V$ and $10V$ with $V = 10^{12}$, for fixed $\rho = 20$ and $\xi = 10$. We can notice the fairness in service as the policy achieves a value close to the optimum for large V.

Figures 4b – 4d show CDFs of the pre-buffering and re-buffering quantities for different values of ξ, for fixed $V = 10^{13}$ and $\rho = 20$. We can observe from these figures that the parameter ξ affects the tradeoff between the pre-buffering time and the re-buffering quantities, namely, the number of re-buffering periods and the percentage of playback time spent in re-buffering mode. From the buffering policy, we can see that a larger ξ sets a larger threshold for starting playback resulting in larger pre-buffering times as reflected in Figure 4b. However, with larger pre-buffering times, we have lesser number of re-buffering periods and the playback spends a lesser fraction of the time in re-buffering mode, as reflected in Figures 4c and 4d.

Finally, Figures 4e – 4g show analogous CDFs for different values of ρ with $V = 10^{13}$ and $\xi = 25$. We can observe that ρ affects the tradeoff between the percentage of skipped chunks and the buffering metrics. From the buffering policy, we can see that if we choose a larger ρ, then the buffer waits longer for undelivered chunks and accommodates chunks with larger delays. As a result, the percentage of chunks which are skipped decreases, as seen in Figure 4e. Further, by waiting longer for chunks with large delays, the buffering policy allows longer pre-buffering and re-buffering times, as evidenced in Figures 4f and 4g. Note that when ρ is set to ∞, no chunk is skipped as seen in Figure 4e. Thus, ρ can be chosen to be equal to ∞ if one wishes to avoid skipping any chunks altogether. However, the chunk-skipping policy provides every user u the flexibility to skip a small percentage of chunks in order to improve the

streaming performance in terms of the pre-buffering and re-buffering performance metrics. In fact, as is evidenced in Figures 4f and 4g, only a small percentage of outlier chunks have large delays and by skipping them, there is a drastic improvement in the pre-buffering and re-buffering times.

B. Experiment 2

In this case the setup is the same as in Experiment 1 but now we consider a specific user, indicated by u_1, moving slowly across the square grid along the green path indicated in Figure (2), during the 1000 slots of simulation. We fix the parameters $V = 10^{13}$, $\xi = 25$ and $\rho = 50$. For the chosen parameters, we observe that the percentage of chunks which are skipped by the mobile user is 1.3% and the pre-buffering time is 162 time slots. Furthermore, from Figure 5b, showing the evolution of the playback buffer Φ_t over time, we notice that there are no interruptions and the playback never enters the re-buffering mode. The helpers are numbered from 1 to 25, left to right and bottom to top, in Figure 2. In Figure 5a, we plot the helper index providing chunk $k = 1, \ldots, 1000$ vs. the chunk index. We can observe that as the user moves slowly along the path, the DPP policy "discovers" adaptively the current neighboring helpers and downloads chunks from them in a seamless fashion. Overall, these results are indicative of the dynamic and adaptive nature of the DPP policy in response to arbitrary variations of large-scale pathloss coefficients due to mobility.

C. Experiment 3

We assume the same system setup as the previous experiments but now simulate the situation where users alternate between *idle* and *active* phases. Initially at $t = 0$, all users are idle and each user independently starts a streaming session with probability $p = 0.005$ at every slot. Thus, the time for which a user stays idle is geometrically distributed with mean $\frac{1}{p} = 200$ slots. Once a user starts a streaming session, it stays *active* during 1000 video chunks. After finishing the requests, it goes back into the *idle* state and may start a new session after a random geometrically distributed idle time. We run the simulation for 3000 slots for different values of the key parameters V, ξ and ρ. We evaluate the performance using the same metrics as in Experiment 1 but with the following differences: 1) The percentage of skipped chunks and the quality (SSIM) averaged over the delivered chunks are now calculated with respect to the total number of chunks spanning multiple streaming sessions of the user; 2) The initial pre-buffering time (in number of slots) is calculated for each streaming session and the CDF is plotted over all the sessions; 3) The number of times playback gets interrupted and enters *re-buffering* mode is normalized

by the number of streaming sessions of the user; 4) The percentage of time spent in *re-buffering* mode is calculated with respect to the total playback time spanning multiple streaming sessions of the user;

We note that the results are very similar to those observed in Experiment 1. For instance, Figure 5c shows a similar fairness in service as V is increased and Figures 5d and 5e show how the parameter ξ affects a similar tradeoff between the pre-buffering time and the frequency of re-buffering or stall events per streaming session. Finally, Figures 5f and 5g show how the parameter ρ affects a similar tradeoff between the percentage of skipped chunks and the percentage of playback time spent in re-buffering mode.

ACKNOWLEDGEMENT

REFERENCES

[1] V. Chandrasekhar, J. Andrews, and A. Gatherer, "Femtocell networks: a survey," *Communications Magazine, IEEE*, vol. 46, no. 9, pp. 59–67, 2008.

[2] J. Hoydis, M. Kobayashi, and M. Debbah, "Green small-cell networks," *Vehicular Technology Magazine, IEEE*, vol. 6, no. 1, pp. 37–43, 2011.

[3] M. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synthesis Lectures on Communication Networks*, vol. 3, no. 1, pp. 1–211, 2010.

[4] F. Kelly, "The mathematics of traffic in networks," *The Princeton Companion to Mathematics*, 2006.

[5] Y. Yi and M. Chiang, "Stochastic network utility maximisation-a tribute to Kelly's paper published in this journal a decade ago," *European Transactions on Telecommunications*, vol. 19, no. 4, pp. 421–442, 2008.

[6] M. Chiang, S. Low, A. Calderbank, and J. Doyle, "Layering as optimization decomposition: A mathematical theory of network architectures," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 255–312, 2007.

[7] M. Neely, E. Modiano, and C. Rohrs, "Dynamic power allocation and routing for time-varying wireless networks," *Selected Areas in Communications, IEEE Journal on*, vol. 23, no. 1, pp. 89–103, 2005.

[8] L. Chen, S. Low, M. Chiang, and J. Doyle, "Cross-layer congestion control, routing and scheduling design in ad hoc wireless networks," in *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings. IEEE*, 2006, pp. 1–13.

[9] M. Neely and L. Golubchik, "Utility optimization for dynamic peer-to-peer networks with tit-for-tat constraints," in *INFOCOM, 2011 Proceedings IEEE. IEEE*, 2011, pp. 1458–1466.

[10] V. Joseph and G. de Veciana, "Jointly optimizing multi-user rate adaptation for video transport over wireless systems: Mean-fairness-variability tradeoffs," in *INFOCOM, 2012 Proceedings IEEE. IEEE*, 2012, pp. 567–575.
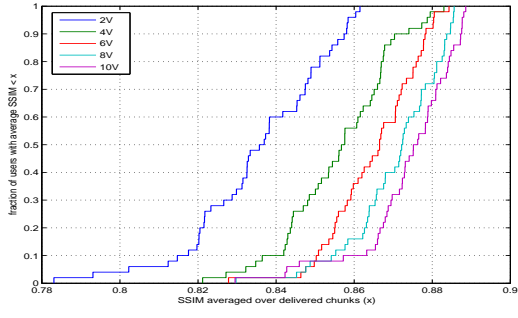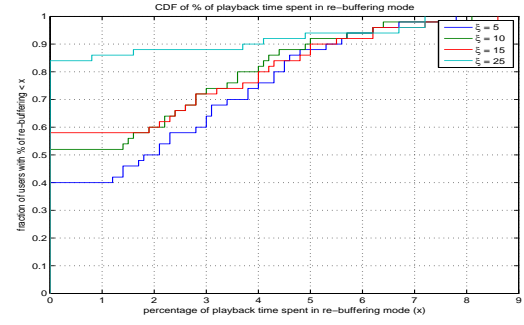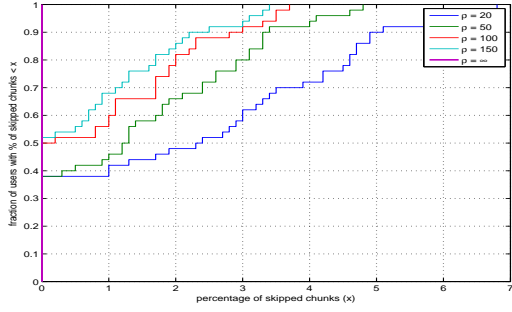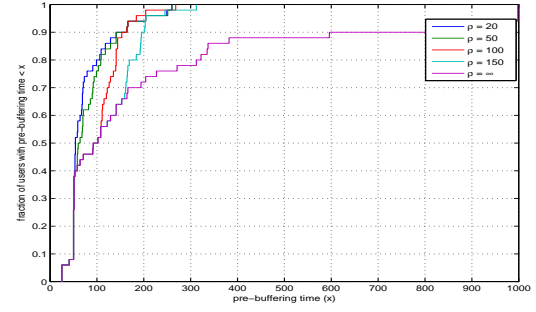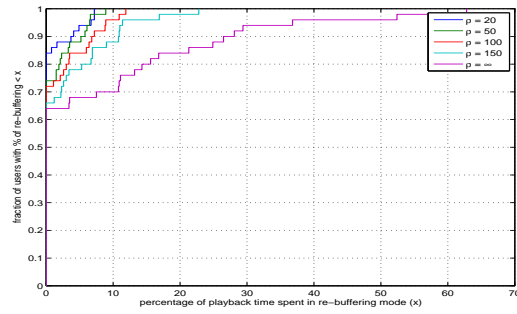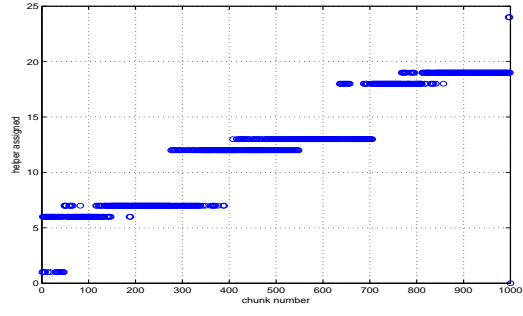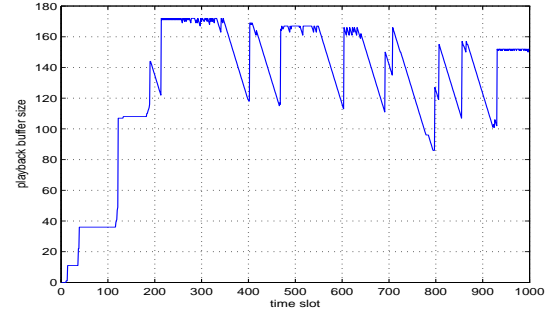
(a)
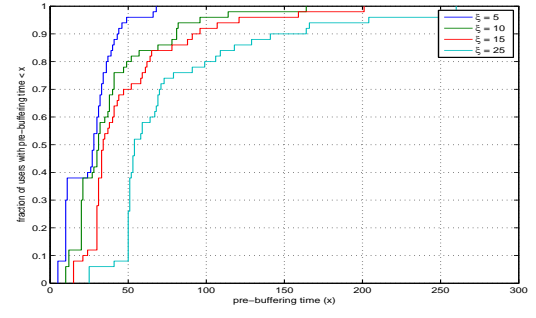
(b)
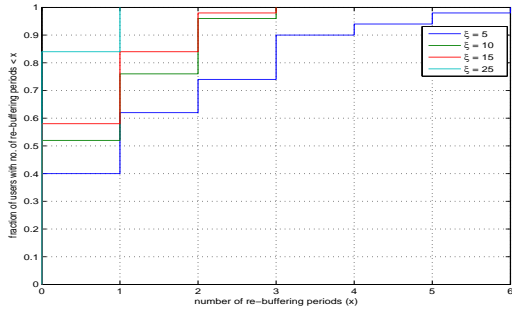
(c)

(d)

(e)

(f)

(g)

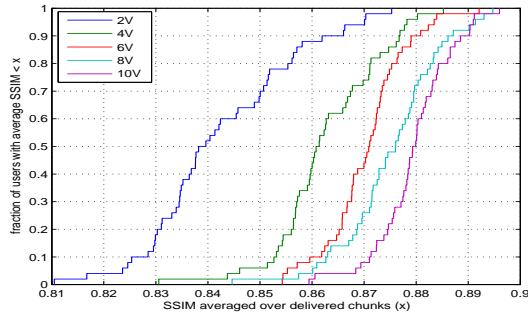Fig. 4: CDF's of different performance metrics for Experiment 1.

[11] C. Chen, R. Heath, A. Bovik, and G. de Veciana, "Adaptive policies for real-time video transmission: A markov decision process framework," in *Image Processing (ICIP), 2011 18th IEEE International Conference on*. IEEE, 2011, pp. 2249–2252.

[12] N. Golrezaei, K. Shanmugam, A. G. Dimakis, A. F. Molisch, and G. Caire, "Femtocaching: Wireless video content delivery through distributed caching helpers," in *INFOCOM, 2012 Proceedings IEEE*. IEEE, 2012, pp. 1107–1115.

[13] J. Wang, C. Yeo, V. Prabhakaran, and K. Ramchandran, "On the role of helpers in peer-to-peer file download systems: Design, analysis and simulation," *Proc. of IPTPS07*, 2007.

[14] X. Wu, S. Tavildar, S. Shakkottai, T. Richardson, J. Li, R. Laroia, and A. Jovicic, "Flashlinq: A synchronous distributed scheduler for peer-to-peer ad hoc networks," in *Communication, Control, and Computing (Allerton), 2010 48th Annual Allerton Conference on*. IEEE, 2010, pp. 514–521.

[15] Y. Sanchez, T. Schierl, C. Hellge, T. Wiegand, D. Hong, D. De Vleeschauwer, W. Van Leekwijck, and Y. Lelouedec, "Improved caching for http-based video on demand using scalable video coding," in *Consumer Communications and Networking Conference (CCNC), 2011 IEEE*. IEEE, 2011, pp. 595–599.

[16] Y. Sánchez, T. Schierl, C. Hellge, T. Wiegand, D. Hong, D. De Vleeschauwer, W. Van Leekwijck, and Y. Lelouedec, "iDASH: improved dynamic adaptive streaming over http using scalable video coding," in *ACM Multimedia Systems Conference (MMSys)*, 2011, pp. 23–25.

[17] A. Ortega, "Variable bit-rate video coding," *Compressed Video over Networks*, pp. 343–382, 2000.

[18] M. Neely, "Universal scheduling for networks with arbitrary traffic, channels, and mobility," in *Decision and Control (CDC), 2010 49th IEEE Conference on*. IEEE, 2010, pp. 1822–1829.

[19] A. Begen, T. Akgul, and M. Baugher, "Watching video over the web: Part 1: Streaming protocols," *Internet Computing, IEEE*, vol. 15, no. 2, pp. 54–63, 2011.

[20] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *Image Processing, IEEE Transactions on*, vol. 13, no. 4, pp. 600–612, 2004.

[21] T. Ho, M. Médard, R. Koetter, D. R. Karger, M. Effros, J. Shi, and B. Leong, "A random linear network coding approach to multicast," *Information Theory, IEEE Transactions on*, vol. 52, no. 10, pp. 4413–4430, 2006.

[22] D. Tse and P. Viswanath, *Fundamentals of wireless communication*. Cambridge Univ Pr, 2005.

[23] A. F. Molisch, *Wireless communications*. Wiley, 2010, vol. 15.

[24] E. Biglieri, J. Proakis, and S. Shamai, "Fading channels: Information-theoretic and communications aspects," *Information Theory, IEEE Transactions on*, vol. 44, no. 6, pp. 2619–2692, 1998.

[25] J. Mo and J. Walrand, "Fair end-to-end window-based congestion control," *IEEE/ACM Transactions on Networking (ToN)*, vol. 8, no. 5, pp. 556–567, 2000.

[26] http://www.tsiwireless.com/docs/whitepapers/LTE%20in%20a%20Nutshell%20-%20Physical%20Layer.pdf.

[27] http://media.xiph.org/video/derf/.

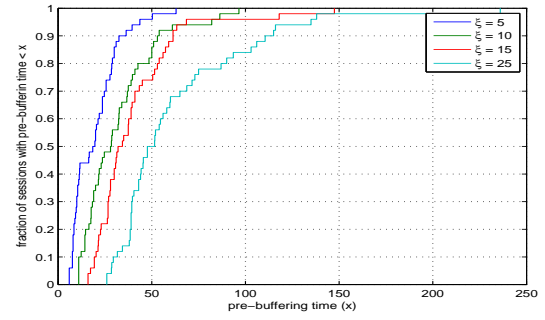[28] https://ece.uwaterloo.ca/~z70wang/research/ssim/.

(a) Seamless downloading of chunks.
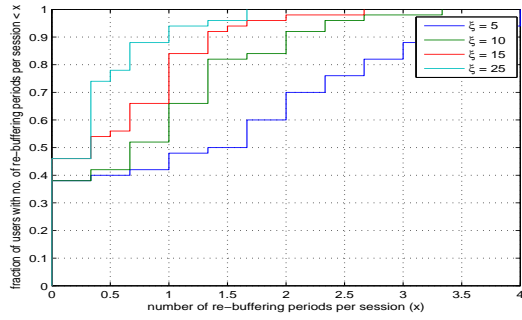

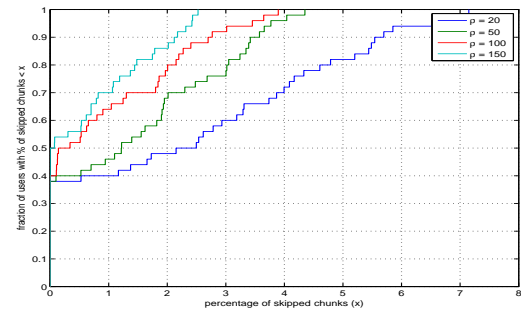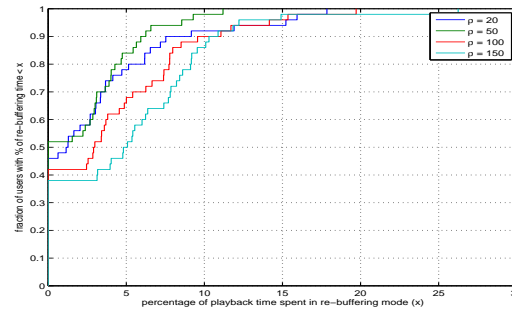(b) Playback buffer dynamics of mobile user.

(c)

(d)

(e)

(f)

(g)

Fig. 5: (a) and (b) correspond to Experiment 2. (c)-(g) indicate the CDF's of different performance metrics for Experiment 3.